Notes on scheduling radio-astronomical observations

Maher Mallem[1] and Frédéric Vivien[1]

Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France
maher.mallem@ens-lyon.fr
frederic.vivien@inria.fr

Keywords: complexity, scheduling, single machine, release date, deadline, astronomy.

1 Introduction

This work tackles the scheduling of radio-astronomical observations, in the context of the Square-Kilometre Array Observatory project (SKAO) and in partnership with the ECLAT laboratory. As an observer from the Earth, one sees celestial bodies move around in the sky over time. The main factor being the Earth revolving around itself, plotting the angle of altitude of a celestial body over time yields a near-sinusoidal signal on a twenty-four hour period. In order to mitigate the impact of the atmosphere of Earth on observation quality, a global altitude angle lower bound is set - typically 20°, at least above the horizon line (0°). Setting such an altitude threshold leads to a daily availability window for each celestial body, which could overlap part of two consecutive days.

Consequently, this setting can be modeled as the makespan minimization of a single machine scheduling non-preemptive instance with a set of n jobs $\mathcal{J} = \{1, \ldots, n\}$, a day length $L \in \mathbb{R}_{>0}$ and, for each job j in $\mathcal{J}$, a processing time $p_j \in \mathbb{R}_{>0}$, a daily release date $r_j \in [0, L)$ and a daily deadline $d_j \in [0, L)$. We denote this problem by $1|\mathit{periodic}_L(r_j, d_j)|C_{\max}$. Surprisingly, to the best of our knowledge, this does not seem to correspond to any previously studied scheduling problem. As such, in this contribution, we propose to investigate the (parameterized) complexity of this scheduling problem, and unveil connections to more established scheduling settings.

Remark 1. W.l.o.g. one can assume that for all jobs j, $p_j \in (0, L]$. Indeed, if $p_j > L$, then the extra multiples of L in p_j correspond to full days of processing job j no matter the starting time of job j. So these extra multiples can be removed from p_j before solving the problem, then they can be inserted back with no issue.

2 Contribution

2.1 Computational complexity

First, we consider the computation of a schedule of minimum makespan OPT. According to the single machine setting with regular time windows $1|r_j, d_j|C_{max}$, our problem is (strongly) NP-complete even on a single day (Lenstra *et. al.* 1977). It even generalizes machine minimization problem $P|r_j, d_j|\min(m)$, where each machine is interpreted as a day. In fact, in our setting, makespan minimization is equivalent to day minimization.

Theorem 1. *If $D \in \mathbb{R}^+$, then decision problem $1|\mathit{periodic}_L(r_j, d_j)|C_{\max} \leq D$ can be reduced to decision problem $1|\mathit{periodic}_L(r_j, d_j)|C_{\max} \leq dL$, where $d = \lceil D/L \rceil$.*

Proof. (Sketch.) If D is a multiple of L, then nothing needs to be done. Otherwise, let $\rho = D \bmod L$. We add a fill job f of processing time ρ, daily release date $(L - \rho)$ and daily deadline 0. Given a schedule τ of makespan at most dL featuring this extra job f,

the sequence of scheduled jobs in τ can be written as AfB. We propose a schedule with job sequence BA for the original schedule. Indeed, job f is necessarily completed at the very end of a day. So the parts Af and B of schedule τ can safely be permuted to obtain a schedule with job sequence BAf and makespan at most dL. Then job f can be removed to yield a schedule of makespan at most L for the original instance. □

In response to the NP-completeness of our problem, we propose the following single-exponential time algorithm.

Theorem 2. $1|\mathit{periodic}_L(r_j, d_j)|C_{\max}$ *can be solved in* $\mathcal{O}(n \cdot 2^n)$ *time.*

Proof. (Sketch.) We propose a dynamic programming algorithm computing the minimum makespan $M(S)$ over all job subsets $S \subseteq \mathcal{J}$. Given a job j and a time t, let $s_j[t]$ be the earliest available start time of job j which is greater than or equal to time t. We set $M(\emptyset) = 0$ and, if $S = \{j_1, \ldots, j_k\}$, then:

$$M(S) = \min_{1 \leq \ell \leq k} (s_{j_\ell}[M(S \backslash \{j_\ell\})] + p_{j_\ell}). \tag{1}$$

The 2^n subsets are considered by nondecreasing order of their size. Each subset takes $\mathcal{O}(n)$ time, with each value $s_{j_\ell}[M(S\backslash\{j_\ell\})]$ taking $\mathcal{O}(1)$ operations - provided that minimum makespan value $M(S\backslash\{j_\ell\})$ has been computed. □

We also show that the preemptive variant of our problem is polynomial-time solvable.

Theorem 3. *Preemptive* $1|\mathit{periodic}_L(r_j, d_j)|C_{\max}$ *is polynomial-time solvable.*

Proof. (Sketch.) Given a makespan threshold $D \in \mathbb{N}$, we provide a translation of the decision problem into a maximum flow problem with $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ edges. First, in $\mathcal{O}(n \log(n))$ time, we compute $u_1, \ldots, u_K$ the increasing sequence of release date and deadline values combined with values 0, (D mod L) and L. Then, on top of source s and sink t, we have two layers of nodes. For every job j in $\mathcal{J}$ we have a node a_j with an edge from source s of capacity p_j. And, for every k in $\{1, \ldots, K-1\}$, we have a node b_k with an edge of capacity $(u_{k+1} - u_k)$ from every job j, the daily time window of which includes interval $[u_k, u_{k+1})$, and an edge to sink t of capacity if and $[(u_{k+1} - u_k) \cdot \lceil D/L \rceil - L]$ if D is not a multiple of L and $u_k \geq (D \bmod L)$, and $[(u_{k+1} - u_k) \cdot \lceil D/L \rceil]$ otherwise.

Given a solution of this maximum flow instance with $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ edges, a schedule can be obtained by greedily filling every daily time interval $[u_k, u_{k+1})$ one day at a time. Finally, a schedule of optimal makespan can be found by binary search on the value of D with initial lower bound $(\sum_{1 \leq j \leq n} p_j)$ and initial upper bound $(nL + \sum_{1 \leq j \leq n} p_j)$ (since there always is an available start time before waiting for a full day). □

Finally, note that our setting is reminiscent of the discrete Interval Scheduling problem, where a collection of arbitrary start times (i.e., not periodic and not necessarily consecutive) is given to each job. While scheduling equal-length jobs with three arbitrary start times per job is (strongly) NP-complete (Keil 1992), our setting can be solved in $\mathcal{O}(n \cdot \log(n)^2)$ time when time windows are tight - i.e., when their length is equal to the job processing time (Dereniowski and Kubiak 2010). This highlights the periodicity of job availabilities as a key property in our setting.

2.2 Approximation

Regarding approximation, the NP-completeness of our problem over a single day implies that there is no better than a 2-approximation algorithm - and thus no PTAS.

Theorem 4. *Unless* P = NP*, for every* $0 < \varepsilon < 1$ *there is no* $(2 - \varepsilon)$*-approximation algorithm for problem* $1|\mathit{periodic}_L(r_j, d_j)|C_{\max}$.

Proof. (Sketch.) Let $\varepsilon > 0$. We reduce from NP-complete problem $1|r_j, d_j|C_{\max} \leq D$ and create a $(2-\varepsilon)$-gap between YES and NO instances. Given an instance $\mathcal{I}$ of the former problem, we set $L = (D+1) \cdot \lceil 1/\varepsilon \rceil$ and create an instance $\mathcal{I}'$ of $1|periodic_L(r_j, d_j)|C_{\max}$ featuring $\lceil 1/\varepsilon \rceil$ disjoint copies of $\mathcal{I}$. For i in $\{1, \ldots, \lceil 1/\varepsilon \rceil\}$, the i^{th} copy fits within daily time interval $[(D+1)(i-1), (D+1)i)$. Now, if $\mathcal{I}$ is a YES instance, then we replicate the solution in every copy to obtain a schedule with makespan at most L. Otherwise, at least one job from the latest copy must be completed after time $2L - (D+1)$. This induces a makespan gap between YES and NO instances which is lower bounded by $(2-\varepsilon)$. □

Now, whenever $OPT \geq L$, we propose a general way to adapt approximation algorithms for the machine minimization setting, at the cost of an extra $2(1 + L/OPT)$ factor.

Theorem 5. *Let $\mathcal{A}'$ be a $f(n)$-approximation algorithm for problem $P|r_j, d_j| \min(m)$ for some function f. Then one can build a $[2(1 + L/OPT) \cdot f(n)]$-approximation algorithm $\mathcal{A}$ for problem $1|periodic_L(r_j, d_j)|C_{\max}$ running in $\mathcal{O}(\max(n, time(\mathcal{A}')))$ time.*

Proof. Let $\mathcal{I} = \langle \mathcal{J}, L, p_j, r_j, d_j \rangle$ be an instance of $1|periodic_L(r_j, d_j)|C_{\max}$. By Remark 1, assume that all jobs j have processing time at most L. Consider instance $\mathcal{I}' = \langle \mathcal{J}, p_j, r_j, d'_j \rangle$ of $P|r_j, d_j| \min(m)$ where $d'_j = d_j + L$ if the time window of job j overlaps with time zero, and $d'_j = d_j$ otherwise. We propose algorithm $\mathcal{A}$ which computes instance $\mathcal{I}'$ from $\mathcal{I}$, computes the schedule τ' returned by algorithm $\mathcal{A}'$ on $\mathcal{I}'$, then proposes the following schedule τ: if τ' is a schedule on m' machines and job j is scheduled on machine $k \in \{0, \ldots, m'-1\}$ in τ', then $\tau(j) = 2kL + \tau'(j)$. Clearly, algorithm $\mathcal{A}$ runs in $\mathcal{O}(\max(n, time(\mathcal{A}')))$ time.

We show that algorithm $\mathcal{A}$ is a $[2(1 + L/OPT) \cdot f(n)]$-approximation algorithm for problem $1|periodic_L(r_j, d_j)|C_{\max}$. Let τ_{OPT} be a schedule for instance $\mathcal{I}$ with makespan OPT. Then we can deduce a schedule τ'_{OPT} for instance $\mathcal{I}'$ on $\lceil OPT/L \rceil$ machines: if job j starts during day $i \in \{0, \lceil OPT/L \rceil - 1\}$, then we set: $\tau'_{OPT}(j) = \tau_{OPT}(j) - iL$. This means that value $\lceil OPT/L \rceil$ is lower bounded by OPT' the optimal number of machines for instance $\mathcal{I}'$.

Now, because schedule τ' uses at most $f(n) \cdot OPT'$ machines, the makespan of schedule τ is at most equal to $2L \cdot f(n) \cdot OPT'$. By the previous paragraph, this value is upper bounded by $2L \cdot f(n) \cdot \lceil OPT/L \rceil$, and thus by $2(OPT + L) \cdot f(n)$. □

Then, a $\mathcal{O}(\log(n))$-approximation algorithm for $1|periodic_L(r_j, d_j)|C_{\max}$ can be inferred from Theorem 5 by considering the $\mathcal{O}(\log(n))$-approximation algorithm for $P|r_j, d_j| \min(m)$ proposed by Cieliebak *et. al.* (2004).

2.3 Fixed-parameter tractability

Finally, we study the fixed-parameter tractability of our problem - i.e., finding parameters k for which there is an algorithm solving instances $\mathcal{I}$ of our problem in $f(k) \cdot poly(|\mathcal{I}|)$ time, where f is a computable function. In this subsection, we suppose that all time values are integers - i.e., day length L, processing times p_j, daily release dates r_j and daily deadlines d_j. By Remark 1, we also assume that all jobs j have processing time at most L.

We consider four parameters: the number of days $\#days$, the width μ - i.e., the maximum number of overlapping time windows -, the slack σ - i.e., the maximum difference between the length of the daily time window of a job and its processing time - and the border flexibility β - i.e., the maximum, over all jobs j, of the minimum between their processing time minus one and their slack plus one. Our problem is para-NP-complete parameterized by each individual parameter (Cieliebak *et. al.* 2004, Hanen and Munier Kordon 2023). We show that our problem is fixed-parameter tractable parameterized by $(\#days + \sigma)$ and $(\mu + \beta)$ by adapting existing fixed-parameter algorithms on the identical

parallel machine setting. In both cases, the main obstacle is to find a way to deal with day-overlapping jobs, i.e., jobs which can be processed over two consecutive days.

Theorem 6. $1|periodic_L(r_j, d_j)|C_{\max}$ *is fixed-parameter tractable parameterized by* $(\mu + \beta)$.

Proof. (Sketch.) We adapt the border schedule enumeration algorithm proposed by Tarhan *et. al.* (2023). Given $t \in\in \mathbb{R}_{>0}$, a border schedule τ at time t is defined as a schedule over a subset of jobs j such that: $\tau(j) < t < \tau(j) + p_j$. Here, we revisit this notion in our periodic setting of period L. We first enumerate over border schedules at daily time $(0 \bmod L)$. Once such a border schedule is fixed, no other job can be processed over multiple days. Thus the rest of the algorithm can unfold in a similar way as in the identical parallel machine setting. This leads to a $\mathcal{O}([2(\mu \cdot \beta + 1)]^{3\mu} \cdot \mu^{\mu+1} \cdot n + n \log(n))$ running time. □

Theorem 7. $1|periodic_L(r_j, d_j)|C_{\max}$ *is fixed-parameter tractable parameterized by* $(\#days + \sigma)$.

Proof. (Sketch.) We adapt the start time enumeration algorithm proposed by Cieliebak *et. al.* (2004) in their section 5.2. In order to deal with day-overlapping jobs, we enumerate over day choices on top of daily start time choices. This leads to a running time in $\mathcal{O}([(\sigma + 1) \cdot \#days]^{(2\sigma+1)\cdot\#days} \cdot n + n \log(n))$. □

3 Discussion

This work gives a first insight on the computational complexity of radio-astronomical observation scheduling. Theoretically speaking, we believe that the main remaining open question is whether there is a constant approximation algorithm for our problem. In fact, Theorem 5 relates it to a corresponding longstanding open question about the machine minimization problem (Cieliebak *et. al.* 2004). And, while insightful, current fixed-parameter algorithms are too slow to be used in practice, notably compared to the single-exponential algorithm proposed in Theorem 2. Going forward, we believe that the model could be refined accordingly to the real-life setting. Radio-observations are geared towards gathering low-frequency signals and thus often take several hours to complete, while switching the target only takes a couple minutes. So day length L could be reinterpreted as a number of daily time steps, i.e., as an integer, possibly given in unary in the input with a reasonably large time step value (e.g., 15min). Furthermore, since observations can usually be split in chunks, as long as the length of each chunk is long enough, some limited preemption could be introduced to the model in order to decrease the minimum makespan value significantly.

References

Cieliebak, M., Erlebach, T., Hennecke, F., Weber, B. and Widmayer, P., 2004, "Scheduling with release times and deadlines on a minimum number of machines". In: *Exploring New Frontiers of Theoretical Informatics*, pp. 209-222, Springer, Boston MA.

Dereniowski, D. and Kubiak, W., 2010, "Makespan minimization of multi-slot just-in-time scheduling on single and parallel machines", *Journal of Scheduling*, Vol. 13, pp. 479-492.

Hanen, C. and Munier Kordon, A., 2023, "Fixed-parameter tractability of scheduling dependent typed tasks subject to release times and deadlines", *Journal of Scheduling*, Vol. 27, pp. 1-15.

Keil, J., 1992, "On the complexity of scheduling tasks with discrete starting times", *Operations Research Letters*, Vol. 12, pp. 293-295.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P., 1977, "Complexity of machine scheduling problems", *Ann. of Discrete Math.*, Vol. 1, pp. 343-362.

Tarhan, I., Carlier, J., Hanen, C., Jouglet, A. and Munier Kordon, A., 2023, "Parameterized Analysis of a Dynamic Programming Algorithm for a Parallel Machine Scheduling Problem". In: *European Conference on Parallel Processing*, pp. 139-153, Springer.